Query Construction and Result Presentation based on Graph Codes

Stefan Wagenpfeil[a], Felix Engel[a], Paul McKevitt[b] and Matthias Hemmje[a]

[a]University of Hagen, Universitätsstrasse 1, D-58097 Hagen, Germany
[b]Academy for International Science & Research (AISR), Derry/Londonderry, Northern Ireland

Abstract

Indexing and Retrieval of Multimedia is generally implemented by employing feature graphs. These graphs typically contain a significant number of nodes and edges to reflect the level of detail in feature detection. A higher level of detail increases the effectiveness of the results, but also leads to more complex graph structures. However, graph-traversal-based algorithms for similarity are quite inefficient and computation intensive, especially for large data structures. Graph Codes provide a flexible and highly efficient solution to deliver fast and effective retrieval. This significant increase in efficiency and effectiveness, especially for Multimedia indexing and retrieval, is applicable to images, videos, text, and Social Media information features. However, existing query construction methodologies and retrieval algorithms have to be extended to utilize the higher level-of-detail and the fusion of these various Multimedia technologies. Thus, in this paper we present detailed concepts of query construction, retrieval, the corresponding user interfaces, result presentation, and a brief overview of our prototypical implementation based on Graph Codes.

Keywords

indexing, retrieval, multimedia, query construction, semantic querying, graph algorithm, graph code

1. Introduction and Motivation

Multimedia assets like images, videos, texts, or audio are deeply integrated in today's life for many users. The ease of creating Multimedia content e.g., on Smartphones, and publishing it on Social Media is unseen in history. Infrastructure services like high-speed networks, cloud-services, or online storage need a good and fast indexing of Multimedia content [1] as e.g. every single minute, 147.000 photos are uploaded to Facebook, 41.6 million Whatsapp messages are sent, or 347.000 stories are posted by Instagram [2]. Users have thousands of pictures on their smartphones and ten-thousands on their computers and storage clouds. As typically, more than one picture is taken from a certain scene, duplicates, similar images, and irrelevant images become more and more. The same arguments can be applied to video, audio, and text information, as well.

For the users, an easy-to-use and highly flexible querying is required, to distinguish between relevant and irrelevant pictures and to provide accurate retrieval results. In our previous work

BIRDS 2021: Bridging the Gap between Information Science, Information Retrieval and Data Science, March, 19th 2021
stefan.wagenpfeil@fernuni-hagen.de (S. Wagenpfeil); felix.engel@fernuni-hagen.de (F. Engel); p.mckevitt@aisr.org.uk (P. McKevitt); matthias.hemmje@fernuni-hagen.de (M. Hemmje)
0000-0003-2100-7589 (S. Wagenpfeil); 0000-0001-9715-1590 (P. McKevitt)

[3][4][5][6], we already introduced a Generic Multimedia Analysis Framework (GMAF) to fuse features from different Multimedia asset types into a single Multimedia Feature Graph (MMFG) and a very effective and efficient mathematical processing model based on 2D representations of the MMFG, called *Graph Codes*. When employing *Graph Codes*, comparison algorithms can be performed with linear complexity (corresponding to the number of nodes and edges), while graph-based algorithms mostly have exponential runtime. A detailed overview of the evaluation is given in [3]. Additionally, *Graph Codes* provide a much higher level-of-detail.

In this paper, we will define the concepts and modeling for query construction and result presentation based on *Graph Codes*. Section 2 summarizes the current state of the art and related works, section 3 contains the detailed model of query construction algorithms and the presentation of their results. Additionally, we built a prototypical implementation to illustrate and refine the concepts. Details of this implementation are given in section 4. Finally, section 5 gives the conclusion and an outlook to future work is given.

2. State of the Art and Related Work

This section provides an overview of current Multimedia feature extraction techniques supporting indexing and retrieval, which either represent or contribute to indexing features of Multimedia content, which are relevant for querying and result presentation. A brief overview and summary of the related works, especially the *Generic Multimedia Annotation Framework (GMAF)*, the *Multimedia Feature Graph (MMFG)*, and the *Graph Code* concept is given as well. Finally, in this section we will illustrate current querying and result presentation techniques.

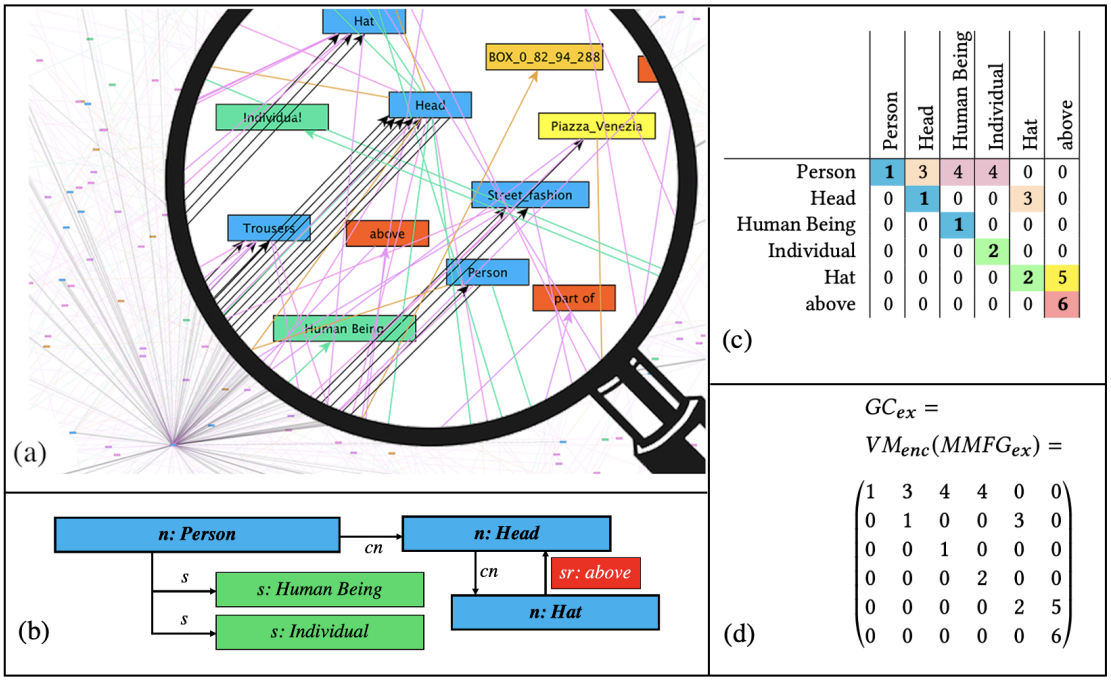
In our previous work, we already introduced the ***Generic Multimedia Analysis Framework (GMAF)*** [3][6] [5][4] as an unifying framework, that is able to fuse various Multimedia features into a single datastructure. The GMAF utilizes selected existing technologies as plugins to support various Multimedia feature detection algorithms for text (e.g. *social media posts, descriptions, tag lines*) [7][8][9], images (especially object detection and spatial relationships including the use of machine learning) [10][11][7][12][7], audio (transcribed to text) [13] [14] [11], and video including metadata [15] and detected features [16][17][14].

The GMAF produces a **Multimedia Feature Graph (MMFG)**, which is defined in [3] and represents various integrated Multimedia features. Within an application, these MMFGs are typically represented as a collection. As the fusion of Multimedia features produces a much higher level-of-detail (LOD), effective and efficient algorithms are required to process these feature graphs. The basis for the definition of these algorithms is a transformation of graphs into another mathematical space for optimized calculations.

Graph Codes are a 2D projection of MMFGs and perform calculations based on matrix algorithms instead of graph traversal algorithms and also support the higher LOD of MMFGs [3]. For these *Graph Codes*, we introduced a metric for similarity calculation, the mathematical concepts of the indexing and retrieval algorithms, as well as a detailed evaluation regarding performance, precision, and recall. This evaluation compares *Graph Codes* algorithms for indexing and retrieval with the corresponding graph-traversal-based algorithms and shows, that *Graph Code* algorithms need linear processing time instead of exponential time for graph-traversal algorithms. Thus, for typical MMIR applications, *Graph Codes* are 5-10 times faster

and due to the higher LOD also more effective in terms of precision and recall. Basically, *Graph Codes* are calculated on basis of an encoded valuation matrix VM_{enc} [18] of a graph (i.e. a valued adjacency matrix). An encoding function f_{enc} calculates a value for each position in the matrix based on the MMFG's feature nodes and edges. Rows and columns of such a *Graph Code* represent the features of a MMFG, and thus the *Graph Code Dictionary*, which employs the labels of each detected MMIR feature.

Figure 1 briefly summarizes this concept as a foundation for the subsequent parts of this paper. Figure 1a. shows a snippet of an exemplary MMFG visualized in a graph editing tool [19], Figure 1b. illustrates a part of the MMFG in an object diagram including various node and relationship types, which can be represented as a *Graph Code* table (see Figure 1c.) based on the graph's valuation matrix. A *Graph Code's* matrix representation is shown in Figure 1d, where the correspondence to mathematical matrix calculations is obvious. It is notable, that due to the current object detection algorithms, MMFGs and their corresponding *Graph Codes* contain feature information (e.g. "is a"), spatial information (e.g. "above"), and temporal information (by the temporal ordering of sub-collections of *Graph Codes*) as well.



	Person	Head	Human Being	Individual	Hat	above
Person	**1**	3	4	4	0	0
Head	0	**1**	0	0	3	0
Human Being	0	0	**1**	0	0	0
Individual	0	0	0	**2**	0	0
Hat	0	0	0	0	**2**	5
above	0	0	0	0	0	**6**

$$\begin{pmatrix} 1 & 3 & 4 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 5 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{pmatrix}$$

Figure 1: Exemplary $MMFG$ and its various representations.

To calculate MMIR results based on *Graph Codes*, a **Graph Code Metric** has been defined, which can be applied for similarity or recommendation algorithms. In general, every detected feature can be regarded as a Multimedia indexing term. The indexing term of any relevant feature thus becomes part of the vocabulary of the overall retrieval index. In Multimedia Indexing and Retrieval (MMIR), these terms typically have structural and/or semantic relationships to each other and represent the basis for query construction and result presentation. In [3], defined a

metric for similary of *Graph Codes*, which is a triple $M_{GC} = (M_F, M_{FR}, M_{RT})$ containing a *feature-metric* M_F based on the vocabulary (i.e. the similarity of common feature vocabulary terms), a *feature-relationship-metric* M_{FR} based on the possible relationships (i.e. the similarity of edges between detected features), and a *feature-relationship-type-metric* M_{RT} based on the actual relationship types (i.e. the similarity of the edge types between features). This metric can be applied for result presentation and has to be considered when constructing corresponding queries.

Current most commonly used **Query Construction** technologies include the employment of structured query languages (e.g. SQL, OQL, XML-Query), as well as Visual query languages (VQLs) and natural language querying (NL). An exemplary summary is given in [20]. A Meaning Driven Data Query Language (MDDQL) [20] [21], can be supportive for query construction by system made suggestions of natural language based terms. In the field of Natural Language Processing (NLP), several approaches have been made, to automatically translate natural language into structured queries, e.g. NLP to SPARQL processing [22] [23]. The Query By Example pattern [24] employes a reference object to define a query for similar objects and the Relevance Feedback interaction templace (including Explicit, Implicit and Pseudo Feedback) [25] can be applied to refine retrieval results. In the remainder of this paper, we will define *Graph Code* query construction models based on these technologies. Typically, results of this kind of queries are represented in form of ranked lists.

In regards of the **Result Presentation**, often ranked lists are used to display results in accordance with various relevance measures [26]. Amongst the most common ones are *similarity*, indicating the best matches for a given query. Result presentation including *recommendations* (indicating connected or corresponding results), *filters* (employed to specify retrieval results), and *streaming* (applied to live data) is typical for MMIR applications. All these technologies provide a profound set of presentation possibilities, but have to be adapted or modified to utilize the full potential of *Graph Codes*.

For **Semantic Querying, Indexing, and Retrieval**, the Resource Description Framework (RDF) and the Resource Description Framework Schema (RDFS) [27] can serve as a foundation. RDF covers the description of any kind of resource by employing XML Syntax, and RDFS can be employed to represent domain specific vocabulary terms as a standardized model for structuring, constructing, integrating, and interlinking semantic representation schemas. Building on RDF and RDFS, querying can be performed by employing e.g., SPARQL, which is a standardized query language and also supports the inclusion of semantic features. As RDF is based on XML, it automatically can be represented in form of a graph model, which gives the opportunity to employ a mapping to the MMFG also on a structural level. RDF-applications like publishing or linking, as well as a shared data model and schema can act as a base layer for other technologies [27] [28] [29] [30]. Our work is designed to closely relate to these existing standards.

In summary, we can state that current technologies provide a sufficient set appropriate algorithms, tools, and concepts for extracting features of Multimedia content. Integrating data structures as, e.g. the MMFG can fuse this information and compile it into a large feature graph structure. However, the need to fuse many features into graphs to increase effectiveness contradicts the demand of higher performance for retrieval, as graph-traversal algorithms become less efficient with an increasing number of nodes and edges. *Graph Codes* can be applied for MMIR to efficiently perform feature-based calculations in a 2D space. Query construction

can utilize existing structured query languages. However, these languages will be not intuitive enough for the user. Therefore, particularly for MMIR applications, an easy-to-use and easy-to-understand query construction is required. In the remainder of this paper, we will illustrate our approach for query construction and result presentation as well as our proof-of-concept implementation for MMIR applications based on *Graph Codes*.

3. Modeling and Design

For the design of our solution, we follow the *User Centered System Design* approach, described by [31] and use UML [32] as a modelling language. The software design follows the GoF-Patterns [33] for reusable software components. For UI/UX-Design we apply the concept of Storyboards [34] to illustrate the interaction between user and application.

Potential relevant Use Cases are illustrated in Figure 2. In this paper we focus on the Use Cases *Manual Query Construction, Query By Example, Query Adaptation* including *Relevance Feedback* for *Query Construction* and the Use Cases *Result Presentation, Recommendation*, and *Filtering* for *Result Presentation.* In the context of this paper, an Asset is a Multimedia object, like an image, video- or audio-file, or connected textual information.

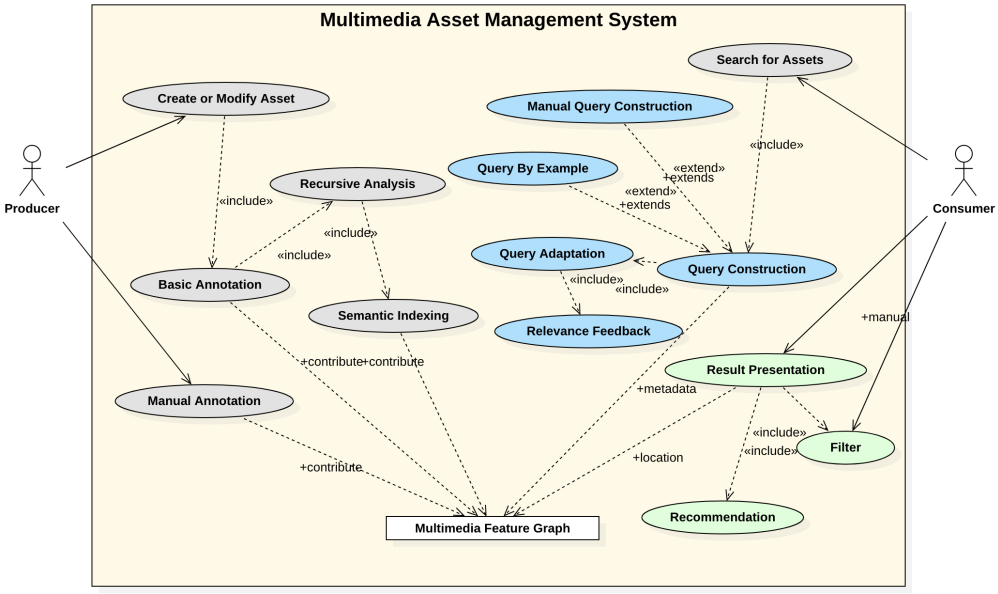


Figure 2: Use Case Diagram for Query Construction and Result Presentation.

For query construction based on *Graph Codes* we identified 3 possible and appropriate options: the application of the *Query by Example* paradigm [24], a manual construction of a query *Graph Code* GC_{Query}, or an adaptation of existing *Graph Codes* including *Relevance Feedback*. For each of these option, a separate user interaction template can be defined, which describes the user interface (including options) and the algorithm for query construction within the MMIR application. In terms of *Graph Codes*, this means, that each of these options somehow has to

calculate a GC_{Query} Graph Code, which is then processed within the GMAF to produce a ranked result list based on the *Graph Code* metrics. In the following subsections, we will describe each of these options.

3.1. Query By Example

Query construction can be based on the *Query by Example* paradigm [24]. In this case, a GC_{Query} is represented by an already existing *Graph Code*, which is typically selected by the user to find similar assets in the collection of a MMIR application. A storyboard of this Use Case is illustrated in Figure 3.

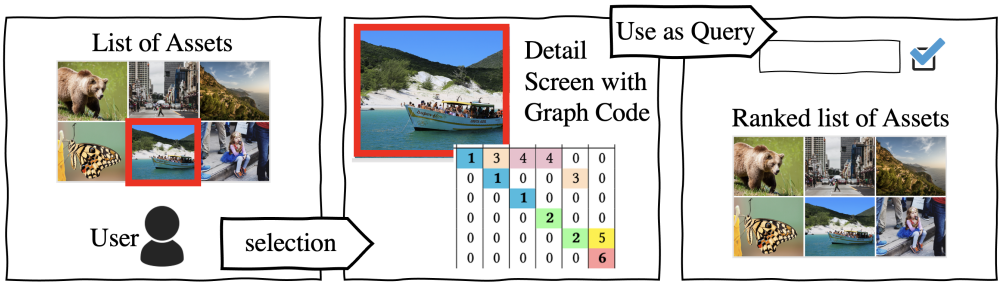


Figure 3: Storyboard for the Use Case *Query By Example*.

In this Use Case, the user can select an asset from the collection of MMFGs. This selection is then represented in detail and in form of its *Graph Code* representation. If the user presses the Button "Use As Query", the *Graph Code* of the selected MMFG is set as GC_{Query}. Then, the query is executed and the list of results is re-ordered according to the similarity between GC_{Query} and each asset in the collection.

3.2. Manual Query Construction

A manual construction of a $MMFG_{Query}$ by users also results in a GC_{Query} *Graph Code*, which then is employed for querying. This manual construction can be performed in various ways, which can be illustrated as follows (see Figure 4):

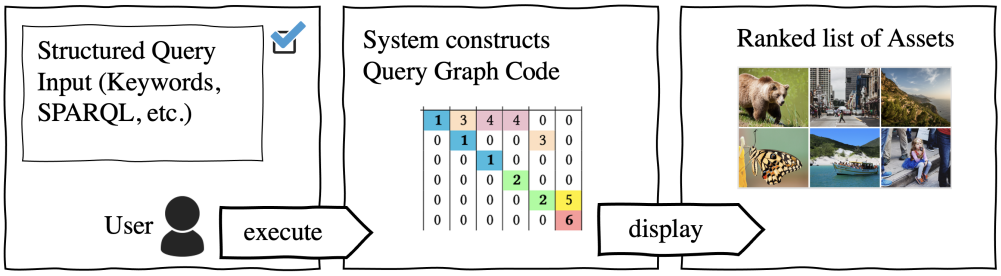


Figure 4: Storyboard for the Use Case *Manual Query Construction*.

In the following, we will describe three modalities of manual query construction: keyword-based querying, structured query languages and natural language processing.

Keyword-based Querying is a very simple manual scenario, in which the user is able to enter keywords into a textfield of the application's User Interface (UI). These keywords will be internally transformed into a *Graph Code*, which is then applied as a GC_{Query} object for query execution. This method is only applicable for queries based on the MMIR application's dictionary (i.e. the set of all *Graph Codes'* vocabulary terms) as a keyword-based entry does not allow the specification of relationships or relationship types. In this case, only the metric M_F can be applied. In many cases, this method will provide an easy-to-use UI and accurate retrieval results. An illustration is shown in Figure 4. However, one major problem of keyword-based querying is, that the user somehow has to have information about valid vocabulary terms of the collection. This can be achieved by presenting a taxonomy [35] or more semantic information [27] to the user. If such a taxonomy is not available, at least a underlying dictionary with synonyms should be employed to map entered terms to valid vocabulary terms. In our modelling, we will address this topic also by *Query Refinement*, which is illustrated in section 3.3.

As a second modality, **Structured Query Languages** can be appleid to *Graph Codes*, as well. As the set of vocabulary terms of an MMIR collection can be represented by RDFS, the user can be also enabled to formulate queries in a structured language (e.g. SQL or SPARQL). In this case, the semantic of the textfield's content can be automatically changed to support a structured query. As structured query languages are based on grammars, an automated detection of the language is possible (e.g. by applying the *Interpreter Pattern* [33]). For each structured query language, the MMIR application has to define a translation into *Graph Codes*. One big advantage of structured query languages is, that relationships and relationship types can be formulated as well and thus also the metrics M_{FR} and M_{RT} can be applied. A SPARQL query for the MMFG shown in Figure 1 can be formulated as follows:

```
SELECT ?x ?y ?z
WHERE {
        ?x rdfs:subClassOf:hat .
        ?x mmfvg:attribute:above .
        ?y rdfs:subClassOf:Person .
        ?y mmfvg:name:Jim .
        ?z mmfvg:type:Image
}
```

Structured Query Input also follows the Storyboard shown in Figure 4. However, entering structured queries is difficult for many users. Amongst others, one common solution for this could be the processing of natural language, which is described in the next subsection.

Thirdly, **Natural Language Processing (NLP)** can be employed to enhance MMIR applications for querying. In many applications, natural language input can enable users to formulate complex queries. Natural language can be entered by writing or in spoken form, which is then typically transcribed by NLP-processors. In any case, the query textfield will contain some kind of natural language query. There are several approaches, that are able to translate NLP into structured query languages like SPARQL [23].

In general, manual query construction will provide an initial GC_{Query}, but in many cases, this query *Graph Code* has to be refined. As this refinement can be applied to any of the so far presented approaches, it will be detailed and summarized in the next section.

3.3. Query Adaptation

An adaptation of an existing *Graph Code* an existing GC_{Query} can lead to a refined GC_{Query} as well. A refinement in terms of *Graph Codes* means, that e.g. some non-zero fields are set to zero, or that some fields get new values assigned according to the *Graph Code* encoding function f_{enc}, or that some new rows and columns are added to the *Graph Codes*. This Use Case can be applied to both *Quer By Example* and *Manual Query Construction* as a refinement.

From a UI perspective, this method requires both the maintenance of a *Graph Code's* dictionary (i.e. its vocabulary terms), and the possibility to modify corresponding relationships and their types. Until now, we employed a table representation for *Graph Codes*, which is a good representation or view. But for manipulating *Graph Codes* in a UI, other methodologies are more suitable and easier to use. Figure 5 shows an optimized workflow for query adaptation, to which all the *Graph Code* metrics (M_F, M_{FR}, M_{RT}) can be applied.

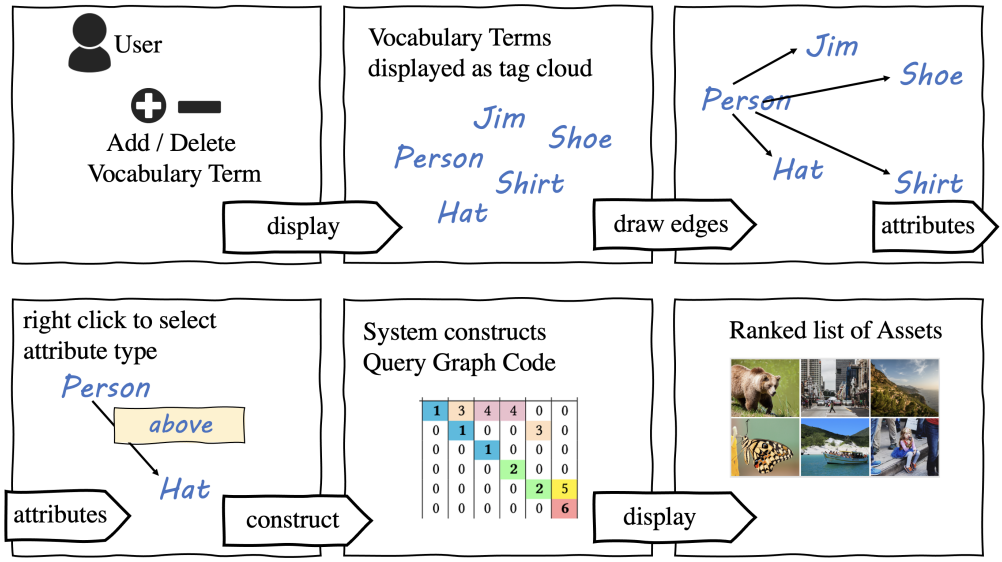


Figure 5: Storyboard for the Use Case *Query Adaptation*.

To adapt or modify a GC_{Query}, whe user can maintain a list of vocabulary terms. These terms can either be suggested by the system based on the underlying taxonomy, thesaurus, or ontology, which would give the user the option to select these terms from a pre-defined list. In this case, only relevant terms of a collection's vocabulary could be applied for query adaptation. But, as *Graph Codes* also include semantic information and relationships to semantic knowledge-graphs like [36], even manually entered terms can be resolved and mapped to entries of the MMIR application's collection. These terms will be used for retrieval based on the metric M_F.

Once a set of terms is defined by the user, the MMIR application provides an option to "draw" relationships between these terms. Any kind of relationship can be utilized by the metric M_{FR}. A very good example for such an UI behaviour is given by yED [19] for graph-based manipulations. Once a relationship is defined between two terms, the MMIR application will ask the user for the relationship-type, which will be relevant for the application of the metric M_{RT}. Once the user has constructed and refined a GC_{Query} in this way, the represented query can be executed by the MMIR application similar to the already described Use Cases.

The options presented in this section are already implemented in the current GMAF prototype for *Graph Code* querying. The current prototype is illustrated in more detail in section 4 and available on Github [5].

3.4. Information Retrieval based on Graph Codes

For information retrieval, we utilize a retrieval function $IR_{GC}(GC_{Query}) = (GC1, ..., GCn)$, which returns a list of *Graph Codes* ordered by relevance implemented on basis of the similarity metric $M_{GC} = (M_F, M_{FR}, M_{RT})$ and thus directly represents the retrieval result in form of a ranked list. It is notable, that the calculation of this ranked list can be performed in parallel, if specialized hardware is available.

For a given query *Graph Code* GC_{Query}, a similarity calculation with each *Graph Code* GC of the collection is performed, based on the *Graph Code* metric M_{GC}. Compared to graph-based operations, matrix-based algorithms can be highly parallelized and optimized. In particular, modern GPUs are designed to perform a large number of independent calculations in parallel [37]. Thus, the comparison of two *Graph Codes* can be done in $O(1)$ on appropriate hardware. Even current Smartphones or Tablets are produced with specialized hardware for parallel execution and ML tasks like Apple's A14 bionic chip [38]. Therefore, the *Graph Encoding Algorithm* also performs well on Smartphones or Tablets.

In [3], we provided detailed facts and figures. To calculate the ranked result list, this algorithm thus utilizes the three metrics M_F, M_{FR} and M_{RT} in a way, that first, the similarity according to M_F (i.e. equal vocabulary terms) is calculated. For those elements, that have equal vocabulary terms, additionally the similarity value of M_{FR} for similar feature relationships is applied for ordering. Also, for those elements with similar relationships (i.e. edges), we also apply the metric M_{RT}, which compares edge types. Hence, the final ranked result list for a GC_{Query} *Graph Code* is produced by applying all 3 *Graph Code* metrics to the collection.

3.5. Result Presentation

There are several options for result presentation. In the context of this paper, we focus on the presentation as a ranked list, as this is the most common form in MMIR applications. Other forms will be part of future work. For presenting the results as a ranked list, the UI has to provide a list of assets. In order to illustrate the similarity values of these assets, the values of each metric should be displayed for each element of the ranked result list. Results are typically presented with the help of preview images and ordered according to their relevance. Relevance itself can be defined in various forms [25], two of the most common forms are initially applied to *Graph Codes*: similarity and recommendations.

Similarity: to show a list of similar assets within a *Graph Code* based MMIR application, the *Graph Code* metric is applied as follows:

```
for each GC in collection
  calculate the intersection matrices of GC_Query and GC
  calculate M_F of GC_Query and GC
  calculate M_FR of GC_Query and GC
  calculate M_RT of GC_Query and GC

  order result list according to
    value of M_F
      value of M_FR where M_F is equal
        value of M_RT where M_F and M_FR are equal
  return result list
```

This algorithm first compares M_F, i.e. the vocabulary terms of two *Graph Codes*. If they have a similar set of vocabulary terms, the metric M_{FR} is applied to determine, if the vocabulary terms are somehow connected to each other. And finally, the metric M_{FR} is employed to check, if the relationship type of the connection between two vocabulary terms is the same. This produces the most similar result to a given GC_{Query}.

Recommendations are retrieval results, that typically extend a result list by including assets, that are somehow relevant for the current query. In case of *Graph Codes*, this can be achieved by applying a different order of metrics to a result list:

```
for each GC in collection
  calculate the intersection matrices of GC_Query and GC
  ...

  order result list according to
    value of M_FR
      value of M_F where M_FR is equal
  return result list
```

This algorithm first employes the metric M_FR, which denotes, that in the asset and the query object, a link between two common features has been found, independent from other matching vocabulary terms. Thus, even if only two vocabulary terms are matching, this metric represents the assumption, that a common link between these terms indicates, that the asset is somehow extending the query object and thus represents some kind of recommendation. Depending on the MMIR application, typically the metric M_F is applied additionally to order these relevance results according to their similarity as well. The metric M_{RT} does not have to be applied for recommendations.

3.6. Summary

In this section we discussed several UI-models for query construction and result presentation. This modeling provides a flexible and extensible design and can be implemented in standalone applications for MMIR, as well as in web applications or for mobile apps. The use cases *Manual*

Query Construction based on keywords, structured query languages or NLP, *Query By Example* utilizing existing similar assets, and *Query Adaptation* providing a highly sophisticated UI for query construction, can be employed to create or refine *Graph Code* queries. The query result will be typically presented as a ranked list. In our current GMAF prototype, we implemented most of the concepts shown in this section. The following section will give a brief exemplary detail on this. Technologies like filtering or streaming will be part of our future work.

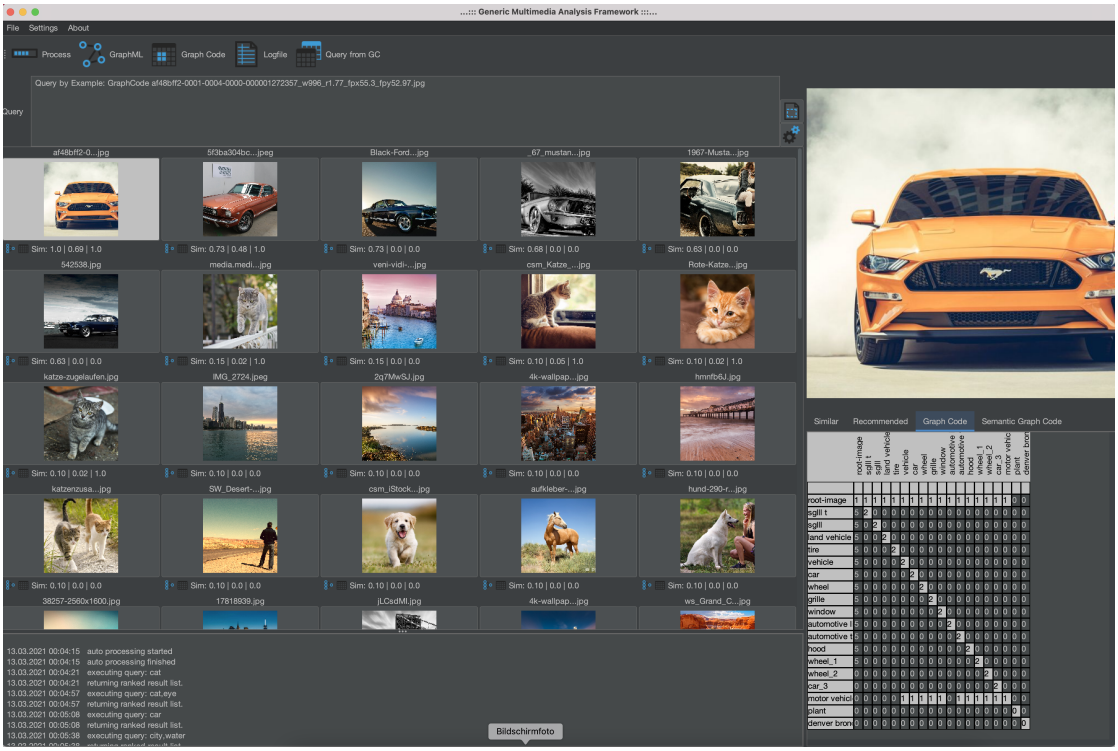


Figure 6: GMAF Prototype for *Graph Code* based MMIR.

4. Implementation

For the implementation of algorithms and concepts, we chose Java [39] as programming languages, as our corresponding frameworks like the *Generic Multimedia Analysis Framework* [6] are already implemented in Java. As the implementation basically follows the algorithms and functions described in the last section, the details of the implementation including source code can be found at Github [5]. In this prototype (see Figure 6), we use Java Swing as a UI language and utilized its existing *Model View Controller* [33] architecture and event model.

For example, the Use Case *Keyword-based Querying* as discussed in section 3.2 has been correspondingly implemented in java as shown in Figure 7. For the implementation of the result presentation in form of a ranked list based on similarity and including additional recommen-

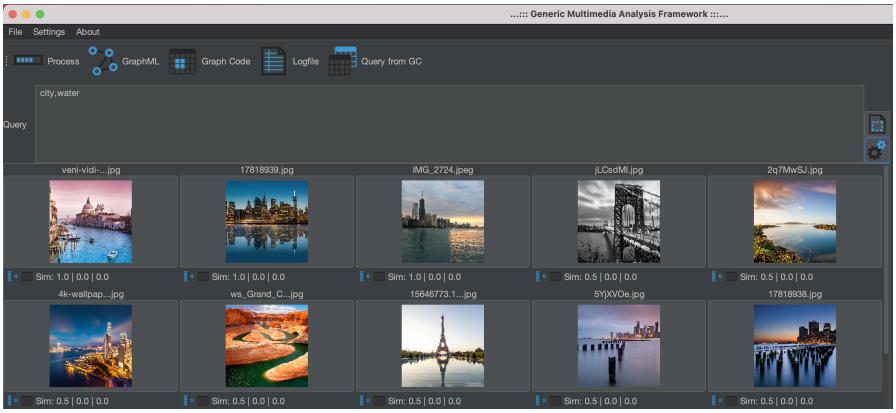


Figure 7: Java implementation of the *Keyword-based querying* and result presentation as a ranked list.

dations for a selected asset For the implementation of similarity and recommendations, we implemented the algorithms as described in section 3. In the UI, we represent similar assets and recommended assets as illustrated in Figure 8.

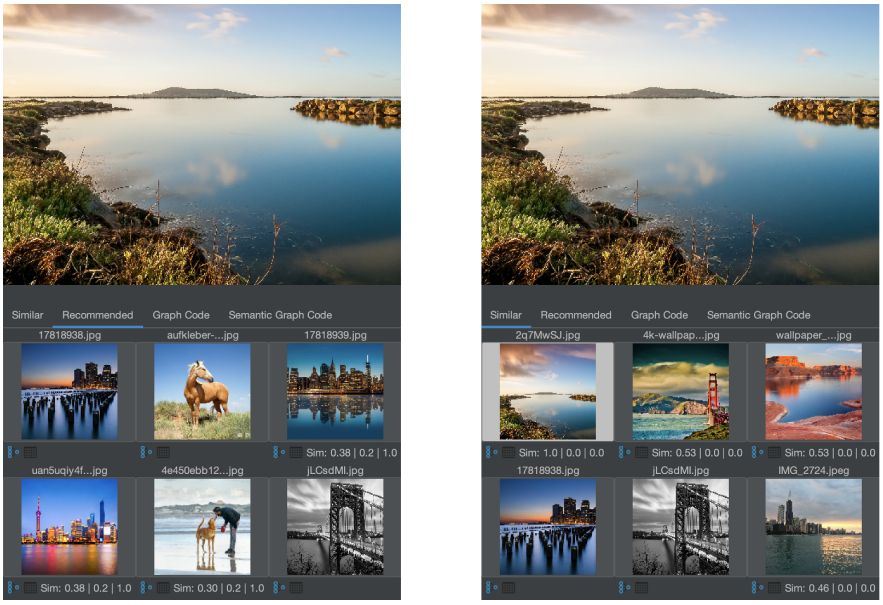


Figure 8: Implementation of recommendations (left) and similarity (right).

Figure 6 shows a complete screenshot of the current *Graph Code* MMIR application. The button "Query from GC" starts a *Query By Example* search based on the selected asset. Below each result, the corresponding metric is displayed for research and refinement purposes and to provide the experienced user a deeper insight into the ranking. The detail view (right section of the screen) shows a preview of the selected asset and additional information like similarity, recommendations, the *Graph Code* and a semantic extension to *Graph Codes*, when a mapping

to existing semantic ontologies can be established. The current prototype is based on images and text only. Further Multimedia asset types are part of our ongoing work. But basically, the UI concepts for *Graph Code* querying and result presentation are completely independent from the Multimedia asset type.

With our current prototypical implementation, we can prove the concept and validity of *Graph Codes* and the corresponding *Query Construction* and *Result Presentation* algorithms. Our implementation shows, that *Graph Codes* can be employed for efficient and effective MMIR. Especially the Use Cases *Manual Query Construction, Query By Example, Query Adaptation*, and *Result Presentation* including *Recommendation* can be implemented based on *Graph Codes* and utilize a high LOD. However, a detailed evaluation has not yet been finalized and will be part of our future work. Currently, only a basic dictionary for english synonyms is employed in the prototype. As part of our ongoing work, we are specifying and implementing semantic extensions to RDF, ontologies, and the semantic web to provide a semantic and intelligent querying as well.

5. Conclusion and Future Work

In this paper we presented a UI modeling for MMIR applications using *Graph Codes*, which significantly improve the level-of-detail, effectiveness and efficiency for MMIR applications. In order to utilize this technology, several aspects of UI design have to be respected. We showed our model for Query Construction and Result Presentation as well as some implementation details and the current UI prototype of the GMAF framework. Open challenges are the complete integration of the Use Case *Query Adaptation* with a highly sophisticated UI, and the support for further Multimedia asset types. Additionally, implementations for mobile devices, tables and implementations as plugins for other existing MMIR applications, will be part of our research and development roadmap based on the *Graph Code* technology.

References

[1] E. Spyrou, D. Iakovidis, P. Mylonas, Semantic Multimedia Analysis and Processing, CRC Press, Boca Raton, Fla, 2017.

[2] J. Clement, Social media - Statistics & Facts, Technical Report, Statista Inc., https://www.statista.com/topics/1164/social-networks/, 2020.

[3] S. Wagenpfeil, F. Engel, P. M. Kevitt, M. Hemmje, Ai-based semantic multimedia indexing and retrieval for social media on smartphones, Information 12 (2021). URL: https://www.mdpi.com/2078-2489/12/1/43. doi:10.3390/info12010043.

[4] S. Wagenpfeil, M. Hemmje, Towards ai-bases semantic multimedia indexing and retrieval for social media on smartphones, SMAP 2020 Conference, 2020.

[5] S. Wagenpfeil, Github repository of gmaf and mmfvg, 2020. URL: https://github.com/stefanwagenpfeil/GMAF/.

[6] S. Wagenpfeil, GMAF Prototype, Technical Report, University of Hagen, Faculty of Mathematics and Computer Science, http://diss.step2e.de:8080/GMAFWeb/, 2020.

[7] J. Beyerer, M. Richter, M. Nagel, Pattern Recognition - Introduction, Features, Classifiers and Principles, Walter de Gruyter GmbH & Co KG, Berlin, 2017.

[8] O. Kurland, J. S. Culpepper, Fusion in information retrieval: Sigir 2018 half-day tutorial, in: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1383–1386. URL: https://doi.org/10.1145/3209978.3210186. doi:10.1145/3209978.3210186.

[9] J. Leveling, Interpretation of coordinations, compound generation, and result fusion for query variants, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, Association for Computing Machinery, New York, NY, USA, 2013, p. 805–808. URL: https://doi.org/10.1145/2484028.2484115. doi:10.1145/2484028.2484115.

[10] A. Bhute, B. Meshram, H. Bhute, Multimedia indexing and retrieval techniques: A review, International Journal of Computer Applications 58 (2012) 35–42. doi:10.5120/9264-3443.

[11] M. S. Lew, N. Sebe, C. Djeraba, R. Jain, Content-based multimedia information retrieval: State of the art and challenges, ACM Trans. Multimedia Comput. Commun. Appl. 2 (2006) 1–19. URL: https://doi.org/10.1145/1126004.1126005. doi:10.1145/1126004.1126005.

[12] C. Hernández-Gracidas, A. Juárez, L. E. Sucar, M. Montes-y Gómez, L. Villaseñor, Data fusion and label weighting for image retrieval based on spatio-conceptual information, in: Adaptivity, Personalization and Fusion of Heterogeneous Information, RIAO '10, Le Centre des Hautes etudes Internationales, Paris, FRA, 2010, p. 76–79.

[13] R. Dufour, Y. Estève, P. Deléglise, F. Bechet, Local and global models for spontaneous speech segment detection and characterization, 2010, pp. 558 – 561. doi:10.1109/ASRU.2009.5372928.

[14] V. S. Subrahmanian, Principles of Multimedia Database Systems -, Morgan Kaufmann Publishers, San Francisco, 1998.

[15] Shih-Fu Chang, T. Sikora, A. Purl, Overview of the mpeg-7 standard, IEEE Transactions on Circuits and Systems for Video Technology 11 (2001) 688–695.

[16] FFMpeg.org, ffmpeg documentation, Technical Report, FFMpeg.org, http://ffmpeg.org, 2020.

[17] X. Mu, Content-based video retrieval: Does video's semantic visual feature matter?, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 679–680. URL: https://doi.org/10.1145/1148170.1148314. doi:10.1145/1148170.1148314.

[18] G. Fischer, Lineare Algebra, Springer Spektrum, 2014.

[19] yWorks GmbH, yEd Graph Editor, Technical Report, yWorks GmbH, https://www.yworks.com/products/yed, 2020.

[20] E. Kapetanios, P. Groenewoud, Query construction through meaningful suggestions of terms, in: J. G. Carbonell, J. Siekmann, T. Andreasen, H. Christiansen, A. Motro, H. Legind Larsen (Eds.), Flexible Query Answering Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 226–239.

[21] E. Kapetanios, D. Baer, P. Groenewoud, P. Mueller, The design and implementation of a meaning driven data query language, 2002, pp. 20– 23. doi:10.1109/SSDM.2002.1029702.
[22] S. M. Hussain, p. kanakam, Transforming natural language query to sparql for semantic information retrieval, International Journal of Engineering Trends and Technology 41 (2016) 347–350. doi:10.14445/22315381/IJETT-V41P263.
[23] H. Jung, W. Kim, Automated conversion from natural language query to sparql query, Journal of Intelligent Information Systems (2020). URL: https://doi.org/10.1007/s10844-019-00589-2. doi:10.1007/s10844-019-00589-2.
[24] I. Schmitt, N. Schulz, T. Herstel, Ws-qbe: A qbe-like query language for complex multimedia queries, in: 11th International Multimedia Modelling Conference, 2005, pp. 222–229.
[25] A. Hamid, Relevance feedback in information retrieval systems (2017).
[26] L. Hassanieh, C. Abou Jaoude, J. Bou abdo, J. Demerjian, Similarity measures for collaborative filtering recommender systems, 2018, pp. 1–5. doi:10.1109/MENACOMM.2018.8371003.
[27] Domingue, John, Fensel, Dieter, Hendler, J. A., Introduction to the Semantic Web Technologies, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 1–41. URL: https://doi.org/10.1007/978-3-540-92913-0. doi:10.1007/978-3-540-92913-0.
[28] R. C. F. Wong, C. H. C. Leung, Automatic semantic annotation of real-world web images, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 1933–1944.
[29] J. Ni, X. Qian, Q. Li, X. Xu, Research on semantic annotation based image fusion algorithm, in: 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), 2017, pp. 945–948.
[30] W3C.org, W3C Semantic Web Activity, Technical Report, W3C.org, http://w3.org/2001/sw, 2020.
[31] D. A. Norman, S. W. Draper, User Centered System Design - New Perspectives on Human-computer Interaction, Taylor & Francis, Justus-Liebig-Universität Gießen, 1986.
[32] M. Fowler, UML Distilled - A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Professional, Boston, 2004.
[33] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns - Elements of Reusable Object Oriented Software, Addison Wesley, 1994.
[34] J. Whalen, Think Human: Customer Centered UX-Design, dpunkt.verlag, 2019.
[35] A. Gilchrist, Thesauri, taxonomies and ontologies - an etymological note, Journal of Documentation - J DOC 59 (2003) 7–18. doi:10.1108/00220410310457984.
[36] Google.com, Google Knowledge Search API, Technical Report, Google.com, http://developers.google.com/knowledge-graph, 2020.
[37] Nvidia.com, Rtx 2080, 2020. URL: https://www.nvidia.com/de-de/geforce/graphics-cards/rtx-2080/.
[38] Wikipedia, Apple a14 bionic, 2020. URL: https://en.wikipedia.org/wiki/Apple_A14.
[39] Oracle.com, Java Enterprise Edition, Technical Report, Oracle.com, https://www.oracle.com/de/java/technologies/java-ee-glance.html, 2020.